#include "include/candy.h"

int main (int argc, char \*argv[])

{

/\* Comprueba los argumentos \*/

if (procesar\_args (argc, argv) == SUCC\_ARGS)

{

return SUCCESS;

}

// Malla malla = {

// .dimens = ver\_params ().dimens,

// .nivel = ver\_params ().nivel

// };

/\* El compilador es una mierda, así que hay que inicializar a mano \*/

Malla malla;

malla.dimens = ver\_params ().dimens;

malla.nivel = ver\_params ().nivel;

/\* Reserva memoria para la matriz y la rellena \*/

reservar\_mem (&malla);

rellenar (&malla);

menu (malla);

return SUCCESS;

/\*

for (int i = 0; i < 10; ++i)

{

mostrar\_malla(malla);

int posY,posX,mov;

printf("PosY: ");

scanf("%d", &(posY));

printf("PosX: ");

scanf("%d", &(posX));

printf("Mov{abajo = 1, arriba = 3, izquierda = 2, derecha = 0}: ");

scanf("%d", &(mov));

\*/

/\*

int bomba;

printf("Bomba: ");

scanf("%d", &(bomba));

int fila = 0;

int columna = 0;

switch(bomba){

case 1:

printf("Fila: ");

scanf("%d", &(fila));

eliminar\_fila(fila,malla);

break;

case 2:

printf("Columna: ");

scanf("%d", &(columna));

eliminar\_columna(columna,malla);

break;

case 3:

recorrer\_malla\_giro(malla);

break;

}

\*/

/\* system("clear");

//Refresh

mover\_diamante(posY,posX,mov,malla);

recorrer\_malla\_coincidencias(malla);

//

system("clear");

mostrar\_malla(malla);

scanf("%d", &(mov));

//

system("clear");

recorrer\_malla\_huecos(malla);

}

return 0;

\*/

}

/\* ---------------- \*/

/\* IMPLEMENTACIONES \*/

/\* ---------------- \*/

/\*

\* es\_valido()

\* Devuelve 1 si el movimineto puede realizarse

\*

\* @param mov

\* Indica el sentido en el que se raliza el movimietno

\*/

int es\_valido(int posY, int posX, int mov, Malla malla)

{

/\*Si devuelve 0 hay error\*/

int b = 0,

cols = malla.dimens.columnas,

filas = malla.dimens.filas;

switch(mov)

{

case 0:

/\*Mover hacia derecha\*/

if(posX < (cols-1)) b = 1;

break;

case 1:

/\*Mover hacia abajo\*/

if(posY < (filas-1)) b = 1;

break;

case 2:

/\*Mover hacia izquierda\*/

if(posX > 0) b = 1;

break;

case 3:

/\*Mover hacia arriba\*/

if(posY > 0) b = 1;

break;

}

return b;

}

/\*\*

\* Realiza un movimiento en la matriz.

\*

\* @param posY

\* Fila del elemento a mover (entre 0 y n-1).

\*

\* @param PosX

\* Columna del elemento a mover (entre 0 y n-1).

\*

\* @param mov

\* Movimiento a realizar (definidos en 'candy.h'):

\* MOV\_DER: 0 -> derecha

\* MOV\_ABAJO: 1 -> abajo

\* MOV\_IZQ: 2 -> izquierda

\* MOV\_ARRIBA: 3 -> arriba

\*

\* @param malla

\* Estructura (definida en 'common.h') con los datos de la matriz de juego.

\*/

void mover\_diamante(int posY, int posX, int mov, Malla malla)

{

Diamante aux;

int cols = malla.dimens.columnas;

Diamante \*matriz = malla.matriz;

if (es\_valido(posY,posX,mov,malla) == 1)

{

switch(mov)

{

case MOV\_DER:

/\*Mover hacia derecha\*/

aux = matriz [(posY \* cols) + posX + 1];

matriz [(posY \* cols) + posX + 1] = matriz [posY \* cols + posX];

matriz [(posY \* cols) + posX] = aux;

break;

case MOV\_ABAJO:

/\*Mover hacia abajo\*/

aux = matriz [((posY + 1) \* cols) + posX];

matriz [((posY + 1) \* cols) + posX] = matriz [(posY \* cols) + posX];

matriz [(posY \* cols) + posX] = aux;

break;

case MOV\_IZQ:

/\*Mover hacia izquierda\*/

aux = matriz [(posY \* cols) + posX - 1];

matriz [(posY \* cols) + posX - 1] = matriz [(posY \* cols) + posX];

matriz [(posY \* cols) + posX] = aux;

break;

case MOV\_ARRIBA:

/\*Mover hacia arriba\*/

aux = matriz [((posY - 1) \* cols) + posX];

matriz [((posY - 1) \* cols) + posX] = matriz [(posY \* cols) + posX];

matriz [(posY \* cols) + posX] = aux;

break;

}

}

else

{

imprimir (DETALLE\_LOG, "Error: movimiento no válido\n");

}

}